**Register Login using Node.js**

**Package to run**

* **Npm i bcryptjs**
* **Npm i jsonwebtoken**

**Config file**

* **JWT\_SECRET=shairy**
* **JWT\_EXPIRE=30d**

**Model =>auth.js**

const mongoose = require('mongoose');

const bcrypt =require('bcryptjs');

const jwt= require('jsonwebtoken');

const userScheme = new mongoose.Schema({

    name:{

        type:String,

        required:[true,'Please enter your name']

    },

    email:{

        type:String,

        required:[true,'Please enter your email'],

        unique:true,

        match:[/^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/,'Please enter valid email']

    },

    password:{

        type:String,

        required:[true,'Please enter your password'],

        minlength:6,

        select:false

    },

    role:{

        type:String,

        enum:['student', 'teacher'],

    },

    resetPasswordToken:String,

    resetPasswordExpire:Date,

    createdAt:{

        type:Date,

        default:Date.now()

    }

});

//Encrypt password using bcrypt

userScheme.pre('save',async function(next){

const salt = await bcrypt.genSalt(10);

this.password = await bcrypt.hash(this.password,salt)

});

//Sign JWT and Return

userScheme.methods.getSignedJwtToken=function(){

    return jwt.sign({id:this.\_id},

        process.env.JWT\_SECRET,

        {expiresIn:process.env.JWT\_EXPIRE})

}

//MAtch user dta with hash password in database

userScheme.methods.matchPassword= async function(enteredPassword){

return await bcrypt.compare(enteredPassword,this.password);

}

module.exports=mongoose.model('user',userScheme);

**Controller=>auth.js**

const ErrorResponse= require('../util/errorResponse');

const asyncHandler =require('../middleware/async');

const user = require('../model/user');

// Insert data into db

exports.register = asyncHandler(async(req,res,next)=>{

    const{name,email,password,role}=req.body;

    const createData = await user.create({

        name,email,password,role

    })

    //CREATE TOKEN

    const token = createData.getSignedJwtToken();

    res.status(200).json({sucess:true,token})

});

//check correct data and then login

exports.login = asyncHandler(async(req,res,next)=>{

    const{email,password}=req.body;

    //validation

    if(!email || !password)

    {

        return next(new ErrorResponse('please provide email and password',400));

    }

    //check user

    const findUser =await user.findOne({email}).select('+password');

    if(!findUser)

    {

        return next(new ErrorResponse('Invalid credential',401));

    }

    //check if password matches

    const isMatch = await findUser.matchPassword(password);

    if(!isMatch)

    {

         return next(new ErrorResponse('Invalid credential',401));

    }

    //CREATE TOKEN

    const token = findUser.getSignedJwtToken();

    res.status(200).json({sucess:true,token})

});

**Router=>auth.js**

const express = require('express');

const{register,login}= require('../controller/auth');

const router = express. Router();

router.post('/register',register);

router.get('/login',login)

module.exports=router;

**Server.js**

const path= require('path');

const express =require('express');

const dotenv =require('dotenv');

const morgan= require('morgan');

const fileupload = require('express-fileupload');

const route=require('./router/router');

const courses=require('./router/courses');

const auth=require('./router/auth');

const connectDB = require('./config/db')

const errorHandler= require('./middleware/error');

dotenv.config({path:'./config/config.env'});

connectDB();

const app=express();

if(process.env.NODE\_ENV==='DEVELOPMENT')

{

    app.use(morgan('dev'))

}

//File upload

app.use(fileupload());

//static folder

app.use(express.static(path.join(\_\_dirname,'public')));

app.use(express.json())

app.use('/api/assignment',route);

app.use('/api/courses',courses);

app.use('/api/auth',auth);

app.use(errorHandler);

const port=process.env.PORT || 5000;

const server=app.listen(port,console.log(`server running in

${process.env.NODE\_ENV} mode using port  ${port}`));

process.on('unhandledRejection',(err,promise)=>{

    console.log(`Error : ${err.message}`);

    server.close(()=>process.exit(1));

})

**Cookies using Node.js**

**Package to run**

* **Npm i cookie-parser**

**Config file**

**JWT\_COOKIE\_EXPIRE=30**

**Controler.js**

//Get token from model, crete cookie and send response

const sendTokenResponse = (User,statusCode,res)=>{

  const token = User.getSignedJwtToken();

  const option ={

      expires: new Date(Date.now()+ process.env.JWT\_COOKIE\_EXPIRE \* 24\*60\*60\*1000),

      httpOnly:true

  };

  if(process.env.NODE\_ENV==="production")

  {

    option.secure=true;

  }

  res.status(statusCode)

  .cookie('token',token,option)

  .json({sucess:true,token});

}

exports.getMe =asyncHandler(async(req,res,next)=>{

    const User =await user.findById(req.user.id);

      res.status(200).json({sucess:true,data:User});

});

**In place of create token use this**

sendTokenResponse(User,200,res);

**complete router page of auth**

const express = require('express');

const{register,login,getMe}= require('../controller/auth');

const{protect}=require('../middleware/auth');

const router = express. Router();

router.post('/register',register);

router.post('/login',login);

router.get('/me',protect,getMe)

module.exports=router;

**Middlewre auth.js**

const jwt = require('jsonwebtoken');

const asyncHandler = require('./async');

const ErrorResponse = require('../util/errorResponse');

const user = require('../model/user');

const { exists } = require('../model/user');

//Protect routes

exports.protect = asyncHandler(async(req,res,next)=>{

    let token;

    if(

        req.headers.authorization &&

        req.headers.authorization.startsWith('Bearer'))

    {

        token = req.headers.authorization.split(' ')[1];

    }

    // else if(req.cookies.token)

    // {

    //     token =req.cookies.token

    // }

    //MAke sure token exists

    if(!token){

        return next(new ErrorResponse('Not authorize to access ',401))

    }

    try{

       //verify Token

       const decoded = jwt.verify(token,process.env.JWT\_SECRET)

        console.log(decoded);

        req.user = await user.findById(decoded.id);

        next();

    }

    catch(err)

    {

return next(new ErrorResponse('Not authorize to access ',401))

    }

});

//Grant access to specific role

exports.authorize = (...roles)=>{

 return(req,res,next)=>{

     if(!roles.includes(req.user.role))

     {

         return next(new ErrorResponse(`user role${req.user.role} is not authorize tp access the route`,403))

     }

     next();

 }

}

**Main router page for submit**

const express = require('express');

const {getAssignment,

    createAssignment,

    getOneAssignment,

    updateAssignment,

    deleteAssignment,

    photoUpload}=require('../controller/controller');

    const advanceResult= require('../middleware/advancedResult')

    /// include other resource router

const courseRouter= require('./courses');

const assignment = require('../model/assignment');

const router = express.Router();

const{protect,authorize}= require('../middleware/auth')

// Re route inro another

router.use('/:assignmentId/courses',courseRouter)

router.route('/:id/photo').put(protect,authorize('student','teacher'),photoUpload);

router.route('/')

.get(advanceResult(assignment,'courses'),getAssignment)

.post(protect,createAssignment);

router.route('/:id')

.get(getOneAssignment)

.delete(protect,authorize('student','teacher'),deleteAssignment)

.put(protect,authorize('student','teacher'),updateAssignment);

module.exports=router;

**user foreign key of user in assignment like course work on controller.js for crud strict restriction**

exports.createAssignment=asyncHandler(async(req,res,next)=>{

        // add user to body

        req.body.user=req.user.id;

        //check for published assignment

        const publishedAssignment = await assignment.findOne({

            user:req.user.id

        });

        //if user is not teacher only add one assignment

        if(publishedAssignment && req.user.role!=='teacher')

        {

            return next(

            new ErrorResponse(`The user with id ${req.user.id} has already published`,404)

        );

        }

        const createAssignment=await assignment.create(req.body);

        res.status(200).json({success:true,data:createAssignment})

});

exports.updateAssignment=asyncHandler(async(req,res,next)=>{

    let updateAssignment= await assignment.findById(req.params.id);

    if(!updateAssignment)

    {

        return next(

        new ErrorResponse(`assignmnet Not found with id ${req.params.id}`,404)

        );

    }

    console.log(updateAssignment);

    //make sure user is assignmnet owner

    if(updateAssignment.user.toString()!== req.user.id && req.user.role!=='teacher')

    {

        return next(

       new ErrorResponse(`user ${req.params.id} is not authersised`,401));

    }

   updateAssignment= await assignment.findByIdAndUpdate(req.params.id,req.body,{

       new:true,

        runValidators:true

    });

    res.status(200).send({success:true,data:updateAssignment})

});

**Forget Password/ Reset Password**

**Controller auth.js**

// forget passwords

exports.forgotPassword =asyncHandler(async(req,res,next)=>{

    const User =await user.findOne({email:req.body.email});

    if(!User)

    {

         return next(new ErrorResponse('There is no user with email ',404));

    }

    // get reset token

    const resetToken=User.getResetPasswordToken();

   await User.save({validateBeforeSave:false})

      res.status(200).json({sucess:true,data:User});

});

**Model user.js**

const crypto =require('crypto');

//Generte and hash password token

userScheme.methods.getResetPasswordToken=function(){

    //Genertae Token

    const resetToken = crypto.randomBytes(20).toString('hex');

    //Hash Token and set to resetPasswordToken field

    this.resetPasswordToken = crypto

    .createHash('sha256')

    .update(resetToken)

    .digest('hex');

    // set expiere

    this.resetPasswordExpire= Date.now()+10\*60\*1000;

    return resetToken;

};

**Router auth.js**

router.post('/forgotPassword',forgotPassword)

**model user,js**

/Encrypt password using bcrypt

userScheme.pre('save',async function(next){

    if(!this.isModified('password'))

    {

        next();

    }

const salt = await bcrypt.genSalt(10);

this.password = await bcrypt.hash(this.password,salt)

});

**Email Sending with password Token**

* **Package node mailer will be used**
* **Use mailtrap.io**
* **npm i nodemailer**

**Conig**

**SMTP\_HOST=smtp.mailtrap.io**

**SMTP\_PORT=2525**

**SMTP\_EMAIL=a0acfe60eac813**

**SMTP\_PASSWORD=eea7da82bb59a5**

**FROM\_EMAIL=er.shairykalra@gmail.com**

**FROM\_PASSWORD=Shairy**

**Util =>send email**

const nodemailer = require("nodemailer");

const sendemail=async (options)=>{

  let transporter = nodemailer.createTransport({

    host: process.env.SMTP\_HOST,

    port: process.env.SMTP\_PORT,

    auth: {

      user: process.env.SMTP\_EMAIL,

      pass: process.env.SMTP\_PASSWORD

    },

  });

  // send mail with defined transport object

  let message = {

    from: `${process.env.FROM\_NAME} <${process.env.FROM\_EMAIL}>`, // sender address

    to: options.email, // list of receivers

    subject:options.subject, // Subject line

    text: options.message // plain text body

  };

  const info = await transporter.sendMail(message);

  console.log("Message sent: %s", info.messageId);

}

module.exports = sendEmail;

**controller auth.js**

const sendemail=require('../util/sendemail')

// forget passwords

exports.forgotPassword =asyncHandler(async(req,res,next)=>{

    const User =await user.findOne({email:req.body.email});

    if(!User)

    {

         return next(new ErrorResponse('There is no user with email ',404));

    }

    // get reset token

    const resetToken=User.getResetPasswordToken();

   await User.save({validateBeforeSave:false})

   //create reset url

   const resetUrl=`${req.protocol}://${req.get('host')}/api/auth/resetpassword/${resetToken}`

   const message = `you are receving email becase someon request for reset password plase make a put request

   in \n \n ${resetUrl}`;

   try {

       await sendEmail({

           email:User.email,

           subject:'password reset token',

           message

       });

        res.status(200).json({sucess:true,data:'email sent'});

   } catch (err) {

       console.log(err);

       user.ResetPasswordToken =undefined;

       user.ResetPasswordExpiere=undefined;

       await User.save({validateBeforeSave:false})

      return next(new ErrorResponse('Email couldnot send ',500));

   }

      res.status(200).json({sucess:true,data:User});

});

**Reset Password**

exports.resetPassword =asyncHandler(async(req,res,next)=>{

    //get hashed token and crypto for that

    const resetPasswordToken=crypto.

    createHash('sha256')

    .update(req.params.resettoken)

    .digest('hex');

    const User =await user.findOne({resetPasswordToken,

        resetPasswordExpire:{$gt:Date.now()}

        });

    if(!User)

    {

         return next(new ErrorResponse('Invalid Token ',400));

    }

    //set neew password

    User.password=req.body.password;

    user.ResetPasswordToken=undefined;

    User.ResetPasswordExpire-undefined;

    await User.save();

     sendTokenResponse(User,200,res);

});